Tampere University

Markus Ryöti

TRAINING BASED TESTING OF TEMPERATURE SENSORS IN DIESEL ENGINE AFTERTREATMENT SYSTEM

Faculty of Engineering and Natural Sciences
Masters of Science Thesis
October 2020

ABSTRACT

Markus Ryöti: Training Based Testing Of Temperature Sensors in Diesel Engine Aftertreatment System
Master's Degree
Tampere University
Mechanical Engineering
October 2020

The goal of this thesis is to research the possibility for creating an automated test for verifying the correct installation of temperature sensors in a diesel exhaust aftertreatment system. Both model and training based solutions are discussed. The implementation is done by training multiple machine learning models and comparing their results.

Data was specifically collected for this research. Data collection consisted of taking measurements from a test engine using different sensor combinations. In total there were four different sensor combinations and for each combination 25 recordings were taken. The recordings had variating starting temperatures. The data was then labeled accordingly for supervised learning.

Results show that classification of sensor installations can be achieved with high accuracy. All the used models provide promising results while logistic regression model seems perform the best. More important and limiting issue is the data gathering process for training and testing the models.

Keywords: machine learning, classification, diesel engine, fault diagnostics

PREFACE

This thesis was written in 2020 at Agco Power in Linnavuori. First of all I would like to thank Dr. Roel Pieters and Dr. Heikki Huttunen for guiding me throughout the process. Their feedback was extremely helpful and I would not have been able to finish my work in such time without their help.

I'm really grateful for Agco Power for providing me this research topic. I also need to thank everyone working at the company for their help and encouragement. Your help has been greatly appreciated.

Finally, I would like to thank my family, friends and my girlfriend for listening, pushing and supporting me during this work.

Tampere, 26.10.2020

Markus Ryöti

TABLE OF CONTENTS

ABBREVIATIONS AND SYMBOLS

AUC	Area Under Curve
CAN	Controller Area Network
CO_2	Carbon Dioxide
DOC	Diesel Oxidation Catalyst
DPF	Diesel Particulate Filter
EAT	Engine Aftertreatment System
ECU	Electronic Control Unit
EGT	Exhaust Gas Temperature
HC	Hydrocarbons
KNN	K-nearest-neighbors
LNT	Lean NO_x Trap
NH_3	Ammonia
NO_2	Nitrogen Oxide
NO_x	Nitrogen Oxides
PM	Exhaust Particle Matter
ROC	Receiver operating Charactheristic
RPM	Revolutions Per Minute
SCR	Selective Catalytic Reduction
SG	Savitzky-Golay
SVM	Support Vector Machine

1. INTRODUCTION

Reliability is an important factor in industrial products. Traditional way of dealing with faults is using breakdown maintenance where the faults and the damages caused by them are fixed when they occur. A more sophisticated and modern way is to strive for preventive maintenance where faults are fixed before they occur. [1]

Emission reducing is an essential task of modern diesel engines. Number of temperature sensors are required in the exhaust aftertreatment system (EAT) to control the conversion of nitrogen oxides (NO_x) and other harmful gases. A lot of times these temperature sensors can be identical to each other in terms of physical appearance which creates a possibility to connect them incorrectly if the installation is not done with caution. These installation mistakes can happen e.g. in the production line or when the vehicle is being maintained in a dealer. If the sensor installation is incorrect, the system will not function appropriately, since proper control cannot be achieved by the Electronic Control Unit (ECU) software. However, the incorrect installation can still allow the engine to be started and the vehicle may be able to operate for a period of time before the fault is detected. At that point, fixing the installation requires significantly more work than it would have taken at the time the sensors were first connected. This will then lead to excessive downtime of the machine causing the machine owner losses from operations and creating a poor customer experience overall.

The purpose of this thesis is to research the possibility to prevent the described situations. The focus is on engines used in off-road applications such as tractors. The diesel engine system that the research is applied to consists of four temperature sensors and a complete EAT system presented in Chapter 2.3. We attempt to solve the problem with first-order model and machine learning based methods. The research is done in Agco Power where there is an existing in house developed diagnostics software tool that is used with a PC. An ideal end-solution would be an automated test integrated into the diagnostics tool. The diagnostics tool user would then start the test and as a result the tool would verify if the sensor installation is done correctly or not. The output would also specify which of the sensors are connected incorrectly if needed.

1.1 Research Questions

This thesis aims to answer the following research questions:

1. Is it possible to validate the correct installation of EAT system temperature sensors based on the measurements?

2. How can the sensor installations be validated based on the measurements they give?

1.2 Research Strategy

The main research strategy of the thesis is empirical research. Research focuses on collecting data of both correctly and incorrectly installed sensors, creating models from those measurements and doing conclusions based on that. Literature review is also done to find similar problems from existing research related to the topic. During the process, existing knowledge in the company was also utilized. There was a lot of existing data of correctly installed sensors and therefore a solid understanding on the behavior of each of the temperature sensor reading in a correctly installed system. This was an advantage that made it more clear on how to problem could be approached.

2. BACKGROUND

This chapter briefly introduces the related engine system components and other relevant matters to the problem. Diesel engine and their emission systems are not presented in full detail, but the essential principles are covered.

2.1 Diesel Engine

Diesel engines are favored in many applications since they provide high thermal efficiency with power and durability [2]. They are used extensively in both passenger cars and off-road vehicles. The applications of the specific engine used in this thesis vary from tractors and combine harvesters to power generators.

Figure 1 *Agco Power HD 74, the engine used in this thesis [3]*

2.2 Emission Standards

Due to their nature of working, diesel engines produce significant amounts of NO_x emissions and other harmful gases and particles [2]. Current diesel engines greatly differ from older ones, mainly due to more sophisticated emission control systems. This is a result

of ever increasing emission regulations and requirements e.g. decreasing Carbon dioxide (CO_2) and NO_x emissions.

The figure below represents the emission limit development regarding hydrocarbons (HC) and NO_x limits between Stage I-V emission standard engines with power output between 130 and 560 kW [4, 5].

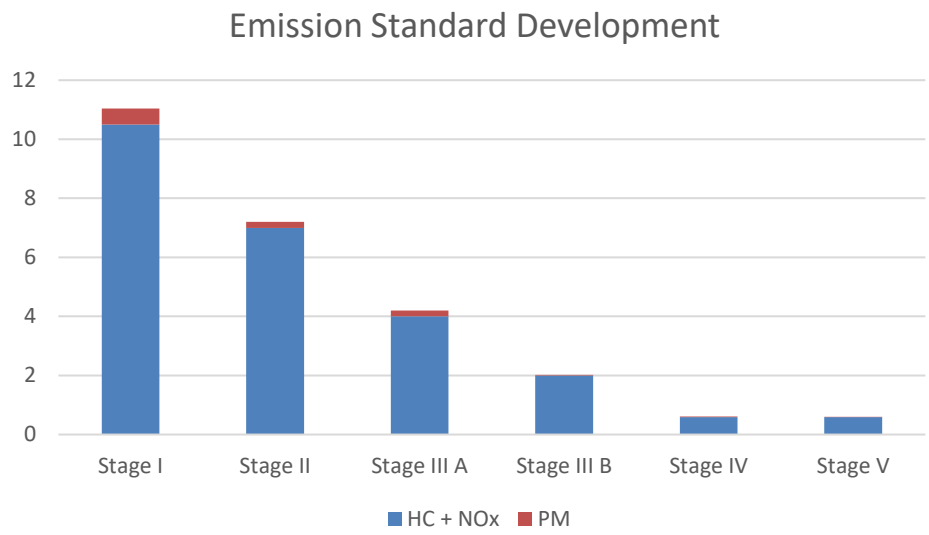

Figure 2 *Emission limits from Stage I to Stage V [4, 5].*

2.3 Engine Aftertreatment

The toxic exhaust gases generated by the diesel engine need to be processed to reduce the amount of harmful gases and particles that are released to the atmosphere. Current Stage V engine aftertreatment systems usually consist of a combination of Diesel Oxidation Catalyst (DOC), Diesel Particulate Filter (DPF) and Selective Catalytic Reduction (SCR) systems.

An overview of the system containing all the components is presented in Figure 3.

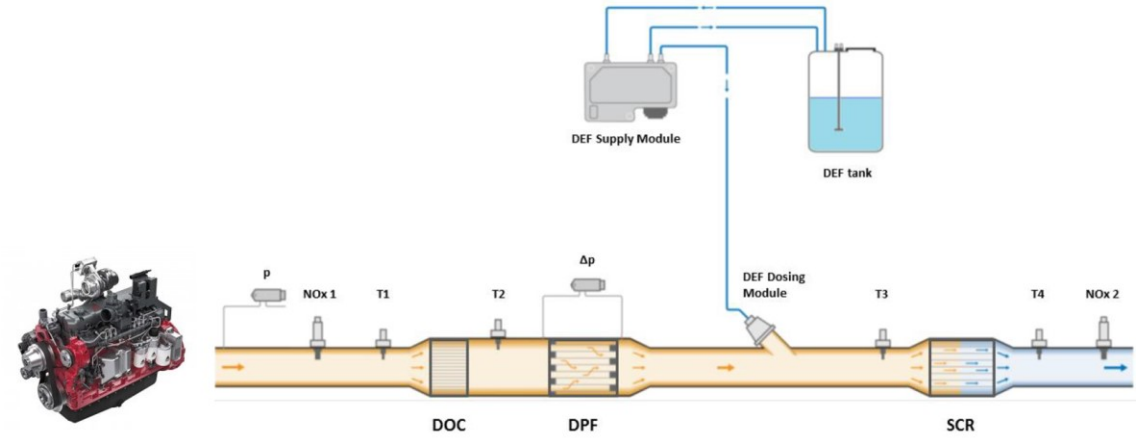

Figure 3 *Stage V aftertreatment system overview [6]*

The temperature sensors (T1-T4) and their locations in relation to other components can also be seen from Figure 3.

2.3.1 Diesel Oxidation Catalyst

DOC is placed before DPF and SCR and it is used to oxidize CO and unburned hydrocarbons in the diesel exhaust system. It also reduces the ratio of Nitrogen Oxide (NO_2) / NO_x and therefore eases the soot and NO_x removal happening in DPF and SCR. Inside the DOC there is a honeycomb cordierite structure which is coated with platinum metal. The structure is packaged in a container made of stainless steel. [7] Temperature sensor 1 is located before DOC and temperature sensor 2 immediately after DOC.

2.3.2 Diesel Particulate Filter

DPF is a ceramic and metallic wall-flow filter which will reduce the amount of particles flowing through. Specifically, DPF is mainly used to collect soot. The collected soot can then be burnt off by a process called regeneration. Regeneration can be done either actively or passively. Active regeneration happens by injecting fuel into the system and therefore increasing the temperature in the DPF. Passive regeneration can be achieved by using different chemical catalysts which reduce the soot oxidation temperature. [7] In the exhaust configuration used in this thesis the DPF is also packaged inside the same metallic container as the DOC.

2.3.3 Selective Catalytic Reduction

SCR system is used to efficiently reduce the amount of NO_x molecules from the exhaust gases. The system injects urea based liquid that is called diesel exhaust fluid (DEF), which will then go through a catalyst. As a result NO_x is converted to non-harmful gases. [7] The positioning of the DEF dosing module in relation to the SCR container can be seen from Figure 3. Temperature sensor 3 is located before the SCR and sensor 4 is located after the SCR.

2.3.4 Emission Reducing Process Overview

The roles of the emission reducing components and their functionalities were described above. The process overview can be visualized with Figure 4:

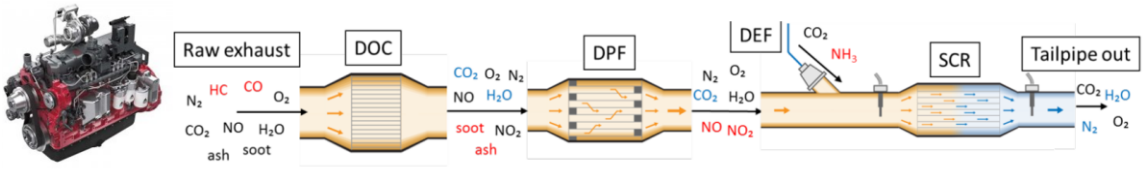

Figure 4 *Emission reduction process overview [6]*

Figure 4 represents the change in particle concentration in the exhaust gases. The chemical reactions happening in the system aren't discussed more in depth since only a surface level understanding is relevant for this thesis.

2.4 Diagnostics Tool

In Agco Power there was an existing diagnostics tool that is used with a PC. The tool has functionalities for example flashing ECU software, viewing sensor measurements, recording engine logs and reading fault codes. There are also functionalities to do other engine related test, e.g. actuator, SCR system emptying and fuel rail pressure tests. If the diagnostics for incorrectly installed temperature sensors would work and could be implemented, a new and similar test feature for validating the sensor installation could be added to the diagnostics tool. An overview of the diagnostics tool is shown in Figure 5.

Figure 5 *Diagnostics tool*

2.5 Existing Studies

There has been multiple studies on model-based and data-driven fault diagnostics in diesel engines. Pezzini *et al.* present a methodology on fault diagnostics of complete Diesel NO_x systems, applicable to both NO_x trap (LNT) or SCR systems. In their study, preliminary analysis is followed by results in simulation system. Their model is able to both detect and also isolate temperature sensor faults from other LNT parametric faults. [8] On similar research Canova *et al.* describe a methodology for model-based sensor fault detection in lean LNT system [9].

Gurung *et al.* describe a method for identifying NO_x sensor faults using histogram-based random forests which is constructed from trucks with both faulty and non-faulty sensors. They state the procedure is generic and reproducible for other components of choice as well. Their model gave around 0.85 AUC in the best case proving the model to be fairly accurate. [10]

Chen *et al.* developed a model-based diagnostic system for detecting dosing and outlet NO_x sensor faults for SCR systems [11]. Similarly Y. Wang *et al.* describe methods for onboard diagnosis for SCR urea injection system [12]. Hu *et al.* put forward a method for failure diagnosis for aged SCR systems with promising simulation results [13]. Guardiola *et al.* present a model-based passive and active diagnostics for DOC oxidation validation again with good results [14].

As described, there has been multiple studies on model based fault diagnosis on diesel engines, their exhaust aftertreatment systems and their different components and sensors. That being said, there's little existing research focusing specifically on EAT system temperature sensor fault diagnostics.

Hu *et al.* Focus specifically on SCR temperature sensors in their study. In that they describe that most SCR temperature sensors faults give three different types of faults. In the first scenario the sensor measurements remain constant. Secondly, measurements will drift linearly from the correct values. In the third form the measurement values are multiplied from the normal values. They present a model based temperature model for fault detection, where sensor measurements are compared to the model estimates. Based on the estimates they evaluate whether the sensors are faulty or not. [2]

3. THEORY

3.1 Model Based Approach

There are many processes and phenomena that can be presented with mathematical models based on how they behave. Therefore a natural and favorable way is to first try to implement such a model for the problem at hand. An example of a model based system is a first order system [15].

3.1.1 First Order System

The temperature sensor reading development in the aftertreatment system could be viewed as a first order system with a step response. The development of such system state is represented in the figure below.

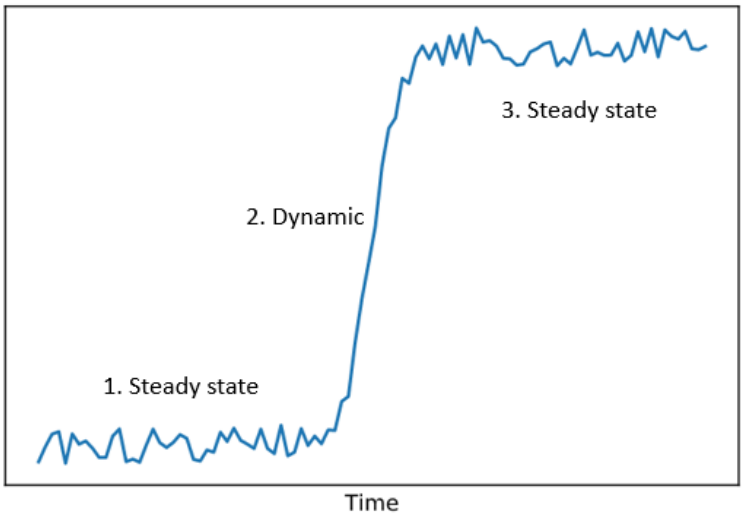

Figure 6 *First order step response overview, adopted from [15, p. 284]*

As can be seen from the figure, a first order model progresses from a starting steady state to a finishing steady state through a dynamic transition phase. The step change occurs when the system is given an input variable *u* which results to a change in the system output variable *y*. More detailed description is presented in Figure 7.

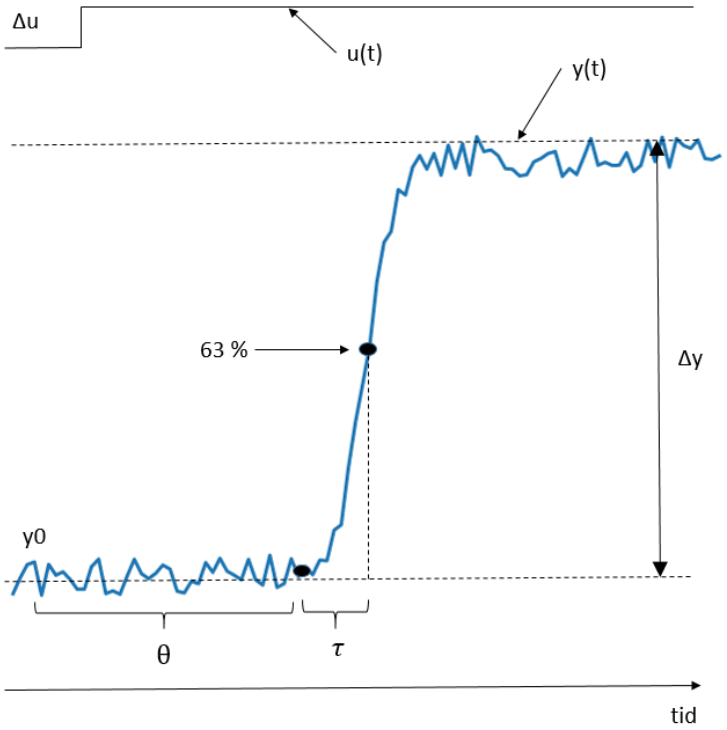

Figure 7 *First order step response variables, adopted from [15, p. 286]*

At the beginning the system is in a steady output y_0. A constant input change of Δu is then given. After the input is applied, there's a delay in output variable change that's marked with θ. The time to reach a specific value in the transformation is marked with τ. Change Δy will occur as a result.

The gain term for the first order system is calculated as a slope for the curve:

$$k = \frac{\Delta y(\infty)}{\Delta u} \tag{1}$$

Where *y* is the output, *u* is the input. The resulting first order step change can be written as:

$$y(t) = y_0 + \left(1 - e^{-t/\tau}\right) k\Delta u \tag{2}$$

Where *y* is the output, *u* is the input, τ is the time constant and *k* is the gain. [15]

3.2 Machine Learning

Machine learning is a mechanism for searching patterns and creating intelligence so that based on its experience the machine will function better in the future. Primary goal of machine learning is to develop an algorithm for a specific problem. [16] In this section some of the essential basics of machine learning are presented.

3.2.1 Supervised And Unsupervised Learning

Machine learning problems can be categorized to supervised and unsupervised learning. In supervised learning the model will not only be given the input data but also a label specifying the output value that the input data will result into. Input can be for example an image and the label is telling what the image contains. Given multiple examples as an input, the model should then be able to evaluate other inputs that are passed in and map those to correct outputs. [17]

In unsupervised learning the labels are not known. In these types of problems, naturally emerging patterns can be examined from the data. This can mean for example clustering where data can be divided into subgroups by their “similarity”. Another example is dimensionality reduction where just the useful input features can be filtered from the data. [17]

3.2.2 Classification And Regression

Supervised machine learning problems can further be divided to classification and regression problems. In classification problems the outputs are categorical and in regression problems the outputs are numerical. [17] Classification task can be for example recognizing whether an image contains a dog or a cat. On the contrary, predicting future stock prices given the historic price data would be a regression task.

3.2.3 Multilabel Classification

In a simple classification the model will assign the pattern or the input to a matching class. Therefore there is only one class per pattern. In binary cases there are two possible classes whereas in multiclass cases there are more than two options. Often there are multiple matching classes for one pattern. For example an image containing a beach

and a mountain would be labeled with both beach and mountain. These types of problems are called multilabel problems in comparison with before mentioned binary and multiclass cases. [18]

3.3 Different Machine Learning Models

In this chapter some traditional machine learning models are briefly described. The main focus in this thesis is classification so all of these algorithms are able to do classification tasks.

3.3.1 K-Nearest-Neighbors

K-nearest-neighbor (KNN) classifier is a memory-based classifier, it will "memorize" where the points it was trained on are located. When the model is given an unseen input, it will find *k* number of nearest training points that are closest to it. Those training points will then vote and the majority vote will then be the classifier output. Though simple, k-nearest-neighbor classifier will often work well and it is often accurate with problems where the decision boundary is irregular. A simple way to create a K-nearest neighbor algorithm is to use Euclidean distance $d_{(i)}$ to measure the difference between data points x:

$$d_{(i)} = \left\| x_{(i)} - x_0 \right\| \tag{3}$$

If there is a situation where the vote is even, the tie is broken at random. [19, pp. 463-483] An example of 3-class classification is shown in Figure 8.

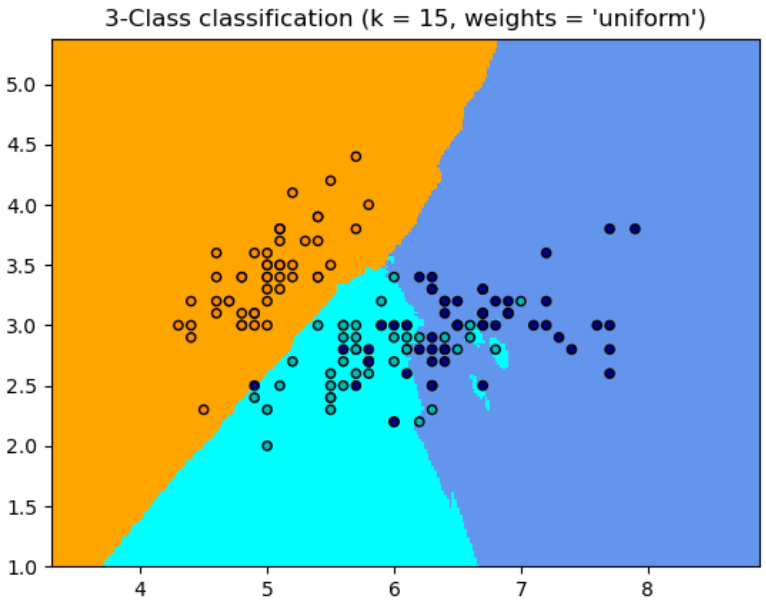

Figure 8 *K-Nearest-Neighbors 3-class classification [20]*

A drawback for the KNN algorithm is its computational load. Finding the neighbors to a data point requires a lot of calculation but also saving the training set to memory causes overhead. [19, p. 480]

3.3.2 Decision Trees

Decision trees are a powerful and widely used machine learning model. As suggested by the name, decision tree algorithm follows a tree-like structure. They have a root node that connects to other nodes by arcs. The tree navigates the arcs until it reaches a leaf node where it makes a decision of the final output. Each node in the tree represents a test of an attribute and branches represent the values of those tests. [16] An example of a simplified decision tree describing the process of predicting whether the weather is good or bad is presented in Figure 9.

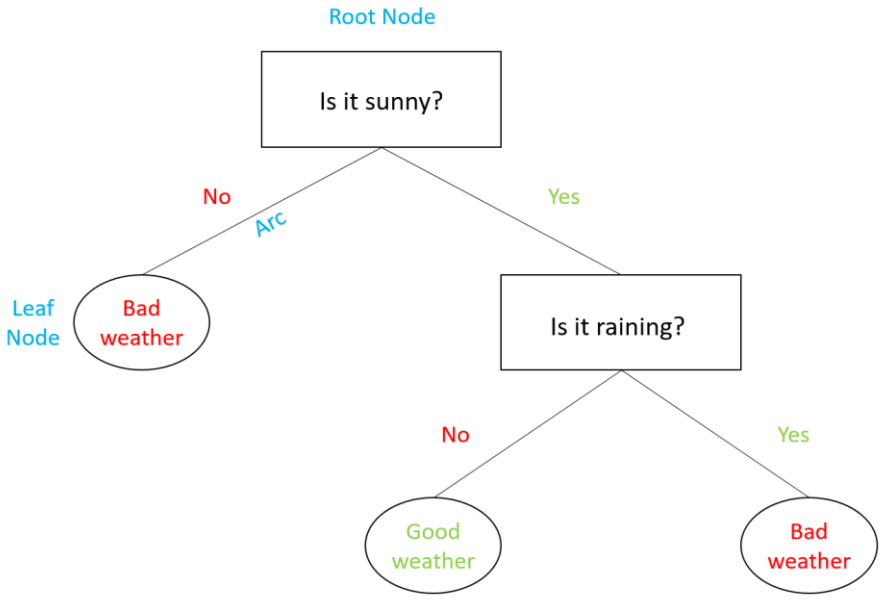

***Figure 9** Decision tree*

Executing a decision three algorithm is fast, in binary cases every node cuts down half of the cases. Trees are also easily understandable and because of that they are sometimes preferred even though more accurate methods would be available. [21]

With classification trees, the goodness of a split in node is measured by its impurity measure. Split is said to be pure if after being splitted for all branches all of the input instances that have selected the branch belong to that class. The probability of class C_i can presented as the following:

$$\hat{P}(C_i|x,m) \equiv p_m^i = \frac{N_m^i}{N_m} \tag{4}$$

Where m is the node, N_m is the number of training instances that reach node m. If the node is pure, the result of the equation will be either 0 or 1. If the node is not pure, the instances should be splitted further to minimize the impurity. [21, pp. 216-220]

3.3.3 Random Forest

Random forest is an ensemble learning method. Ensemble models work by creating multiple classifiers and their output is then combined e.g. with averaging [22, p. 2].

Random forests are an extension of decision trees. They make use of what's called bagging or bootstrap aggregation, which is used for reducing the variance of the prediction functions. Random forest algorithms generate a collection of decision trees that are not correlated and the algorithm then averages those trees. Trees are generally known to be noisy, which makes averaging them beneficial. The correlation reduction between the trees is achieved by randomly selecting the input variables or features while preserving the original number of features. This will most likely allow duplicates and some features are not included in individual trees. With classification, the algorithm gets a vote from all the trees and the classification is then done using the majority vote. This can be presented with the equation:

$$\hat{C}_{rf}^{B}(x) = majority\ vote\left\{\widehat{C_b}(x)\right\}_1^B \tag{5}$$

Where $\widehat{C_b}(x)$ is the predicted class of the *b*th tree. [19 pp. 586-604, 22 pp. 157-174, 23 pp. 330-342]

One useful feature of random forest is the use of out-of-bag (OOB) sampling. This means that for every observation of an input, a predictor is averaged of the trees where that input did not appear. In essence, this makes use of cross validation while being trained. [19]

Random forests are popular and they often do well without tuning. They can also give variable importance values of the features which can help during the feature selection process. [19, pp. 586-604]

3.3.4 Extremely Randomized Trees

Extremely Randomized Trees algorithm is really similar to random forest, but the nodes inside the trees are splitted randomly with random subset of features. The algorithm also uses the full original learning sample compared to Random Forest algorithm were samples are selected randomly and duplicates are therefore allowed. The idea behind Extremely Randomized Trees is to further reduce the variance of the model. [24]

3.3.5 Logistic Regression

Despite the name, logistic regression is a classification algorithm rather than a regression algorithm. At its core, logistic regression is really a one-neuron neural network. Each

input value is associated with a node in this network with its own calculated weight. When the inputs are multiplied their weights and a bias is added, this results into a logit function z. [25] Example is presented in Figure 10.

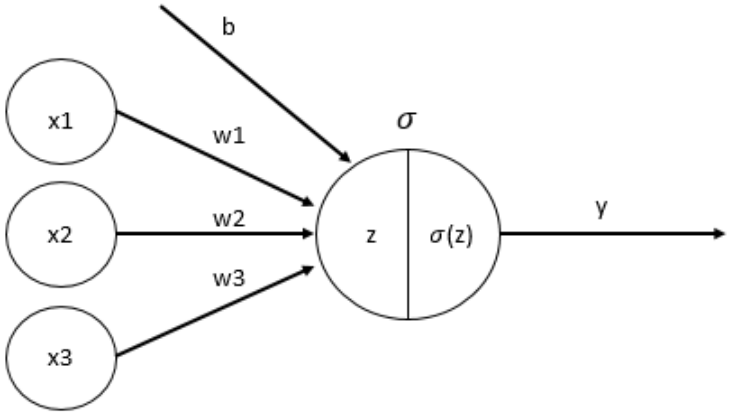

Figure 10 *Logistic Regression schematics, adopted from [25]*

For the figure above the logit function is the following:

$$z = b + w_1 x_1 + w_2 x_2 + w_3 x_3 \tag{6}$$

The resulting logit function is then inputted to a sigmoid function and this will produce the output y.

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{7}$$

The ouput will have a value between 0 and 1. If the value is below 0.5, this means that the input is classified as 0 and conversely if it is above 0.5, the input is classified as 1. During the learning process, the algorithm will find the best fitting weights and bias. In essence, the weights represent the importance of the related feature. [25]

In order to update the weights and bias, an error function is required. An example of this a sum of squared error:

$$E = \frac{1}{2} \sum_n (t^{(n)} - y^{(n)})^2 \tag{8}$$

Where $y^{(n)}$ is a logistic regression output sample and $t^{(n)}$ is the corresponding true label sample. This error is then minimized for a number of iterations called epochs. [25] This will result to best matching weights and bias for the data, therefore increasing the accuracy of the model.

3.3.6 Support Vector Machine

Support vector machines (SVMs) are another tool for classification tasks. The algorithm will fit a hyperplane between the classes that are being classified. The best fitting hyperplane is found when the margin to the separated classes is maximized. [16] Example of a splitting hyperplane or decision boundary is shown in Figure 11.

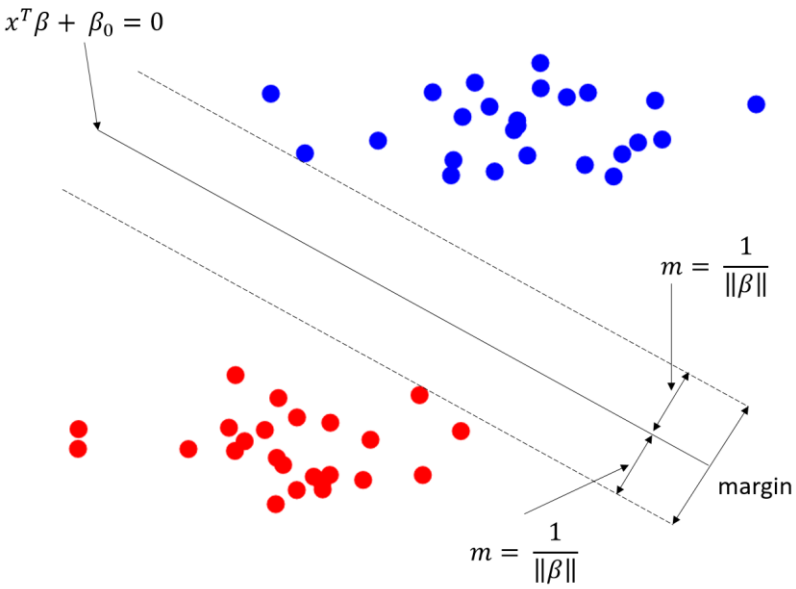

Figure 11 *SVM decision boundary, adopted from [19, pp.417-437]*

Figure 11 presents a separable case where none of the data points are in the margin area. Often there can be overlap with the classes and the case is then non-separable. [19]

A hyperplane can be defined with the following:

$$\{ x: f(x) = x^T\beta + \beta_0 = 0 \} \tag{9}$$

β being a unit vector $\|\beta\| = 1$. In simple separable case creating the biggest margin between the classes can be expressed with the following:

$$\underset{\beta, \beta_0,\ \|\beta\| = 1}{max}\ m \tag{10}$$

$$subject\ to\ y_i\left(x_i^T\beta + \beta_0\right) \geq m, \qquad i = 1, \ldots, N \tag{11}$$

Where m is the margin to the nearest data points between both sides of the decision boundary. [19]

SVMs can also make use of what are called kernel functions. If the problem is nonlinear, the problem can be mapped to a different space by applying a nonlinear transformation function. This can be done efficiently by using a kernel trick. Doing so, the default inner product functions are replaced with kernel functions which means that the dot products are done in the original space and the original instances don't have to be mapped to a new space. [19, 21]

Support vector machines have also proven to be performant in many applications and have been used extensively [19, 21].

3.4 Model Validation

After the model has been trained, it needs to be evaluated somehow. In this chapter some of the basics of model validation are represented.

3.4.1 Bias And Variance

Machine learning models can behave differently with their training data compared to new and unseen testing data. In these situations there are problems with the models bias and variance.

If a model that performs well in training has great error with unseen test data, the problem is its variance. Variance measures how much the model depends on the data it was trained on. In this case the model is said to be over-fitting. The model can also be too simple when it doesn't represent the data accurately enough. In this case the model is said to be under-fitting and the problem is its bias. [16 pp. 27-29, 19 pp. 219-259] Presentation of the correlation between variance and bias in models is shown in Figure 12.

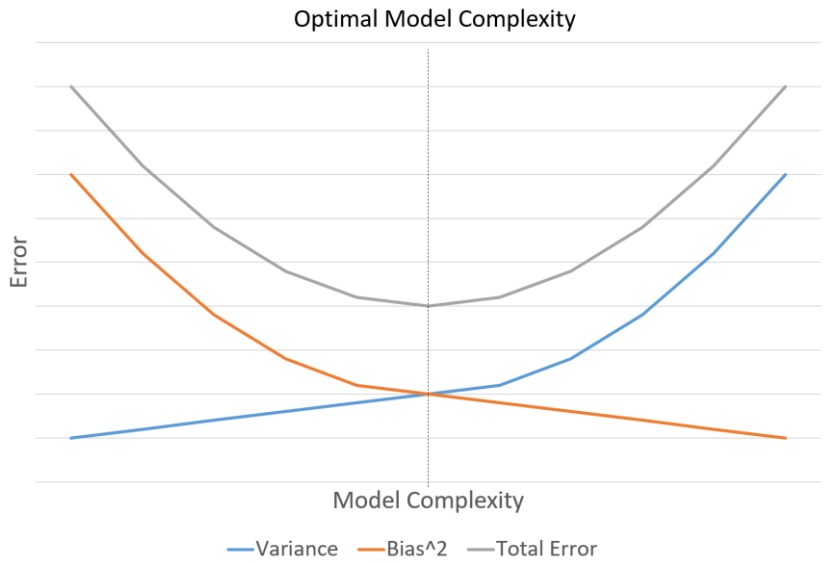

Figure 12 *Bias and variance, adopted from [16]*

In order to have the best functioning model, there needs to be a balance with the models bias and variance. This requires experimenting and finding the best possible parameters during training.

3.4.2 Cross Validation

A simple and often used method for estimating model accuracy is cross-validation. The objective off cross-validation is to measure how the model is able to generalize to different data. With K-fold cross validation method the dataset is randomly split into *K* number of equal sized parts that are called folds. [19, 21] A visual example of 5-fold cross-validation is shown in Figure 13.

Fold 1 (Train)	Fold 2 (Train)	Fold 3 (Train)	Fold 4 (Test)	Fold 5 (Train)

Figure 13 *5-Fold Cross Validation*

In the example above the third fold is reserved for testing and the other four are used to fit the model. The process is repeated so that each of the folds will be used for testing at a time. The sum of all these test results is then averaged to form the final result. [19]

3.4.3 Confusion Matrix, ROC And AUC

With classification, the model's performance can be evaluated better than just looking whether the model classified the input as true or false. This can be done with a help of confusion matrix, example presented in Figure 14.

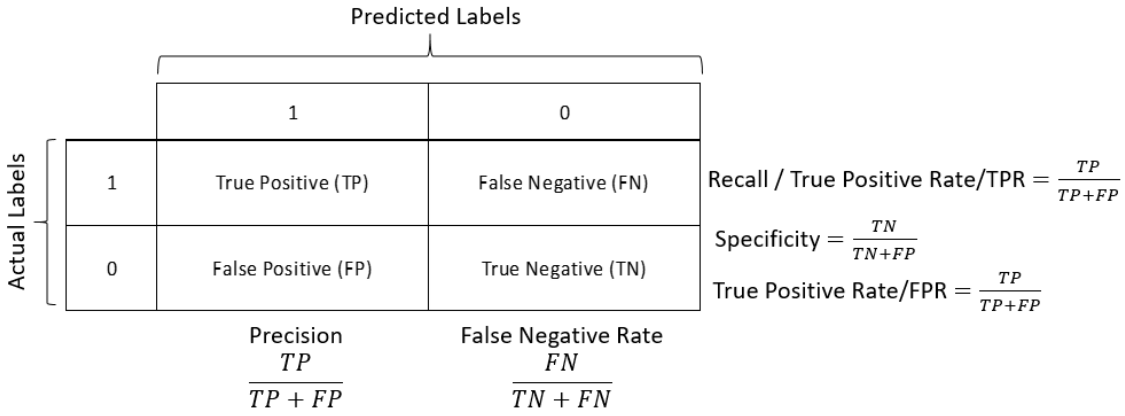

Figure 14 *Confusion matrix for binary classification, adopted from [17]*

In a binary case, the confusion matrix has four cells. The horizontal axis presents the predicted labels and the vertical axis presents the actual labels. With those values, essential indicators can be calculated. Firstly, the precision tells the ratio of correctly classified instances of the total number of positively classified ones. Recall presents the ratio of correctly classified instances of all the existing positives. False negative and false positive rates present the ratio of incorrectly classified instances. [17]

To further visualize the goodness of the classifier performance, ROC or receiver operating characteristics can be used. [17, 21] An example of an ROC curve is shown in Figure 15.

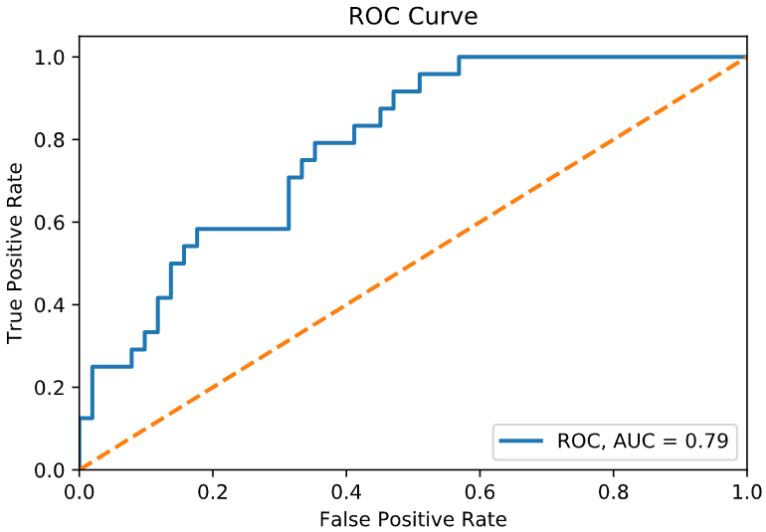

Figure 15 *ROC curve, adopted from [26]*

As can be seen from Figure 15, the ROC curve maps the relationship between the true positive (TP) and false positive (FP) rates. In an ideal case the true positive rate is 1 while the false positive rate is 0. Therefore the better the model performs, the closer its ROC curve is to the top left corner. The diagonal line represents the worst performance hence everything underneath that could be flipped to give better results. The performance of the classifier can be further simplified by calculating the AUC or the area under the curve from the ROC curve. In an ideal case the AUC will result to 1. [21]

3.5 Savitzky-Golay Filter

Real world recorded signals often have noise associated with them. To reduce the amount of noise, different filters can be used. Measurement filtering can also be used to create better features for a machine learning model [27, 28].

Savitzky-Golay or Savgol (SG) filter uses a moving convolution window and a polynomial function is fitted to the points in the window. The polynomial function is fitted inside the window using the least-squares method. [27-30]

A Savitzky-Golay filter that has a length of *N* and polynomial order *d* that is used to smooth noise in sequence *x(n)* at steady state can be expressed with the following:

$$y(n) = \sum_{m=-M}^{M} b_0(-m)\, x\,(n-m) \tag{12}$$

Where b_o is the center of the filter and x is a vector containing the points inside the window. Derivatives can be applied to the polynomials that the Savitzky-Golay filter fits inside the convolution window. When equation 12 is differentiated, the filter output can be generalized:

$$y^i(n) = i! \sum_{m=-M}^{M} g_i(-m)\, x\,(n-m), \qquad i = 0,1,\dots,d. \tag{13}$$

Where i is the derivative order. [27] Example of a noisy sinusoid signal that is filtered with regular SG-filter and derivative SG-filter is shown in Figure 16.

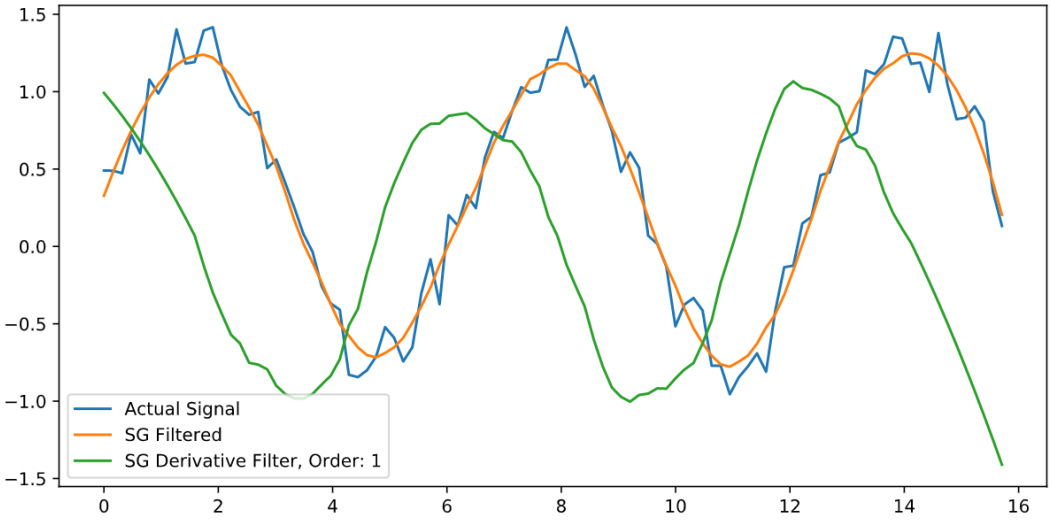

Figure 16 *Savgol filtered sinusoid signals*

The Savitzky-Golay filter is simple yet efficient and it is therefore widely used in different industries [30].

4. IMPLEMENTATIONS

This chapter presents the selected approach, set limitations and discusses the strategies for implementation.

4.1 Test Process

As presented in the beginning, the idea is to have a test that will verify the correct installation of the exhaust temperature sensors. In practice, this would happen by starting the engine and then the test from the diagnostics tool that has a connection to the engine via Controller Area Network (CAN). After that, the test would start to heat up the engine and its exhaust line. There can be two phases in the heatup. In the first phase the engine itself is heated up with constant engine revolutions per minute (RPMs). In the second phase, the RPMs are set to a higher setpoint in order to heat up the DOC. If the engine is already warm at the beginning, then the engine heatup phase will be skipped. After the DOC is heated up, the test will stop and engine will be switched to idle. After that, the test will present a result of which sensors are installed correctly and those that are potentially incorrectly installed.

4.2 First Order System Challenges

While the first order system itself is simple in theory, it faces difficulties when applied in practice to this specific problem. A first order system should start from a steady state and also end to a steady state. Ending state is easy to define by selecting a target temperature and giving the system an input up to that point. However, the starting state is somewhat unclear. In a laboratory setting with predefined starting temperatures the model would likely function fairly robustly. In practice there can be a lot of starting states. Not only can the operator start the test right after cold start or conversely at a point where he or she has been running the machine for several minutes and perhaps with load. Ambient conditions can also vary vaguely depending on the geolocation. Ideally the test would be able to respond to all of these situations.

But even more importantly the catalytic reactions in SCR also create difficulties in a first order model based approach. The temperature changes significantly depending on the state of the catalytic reactions inside the SCR. An example of the temperature changes happening in the EAT because of SCR reactions is presented in Figure 17:

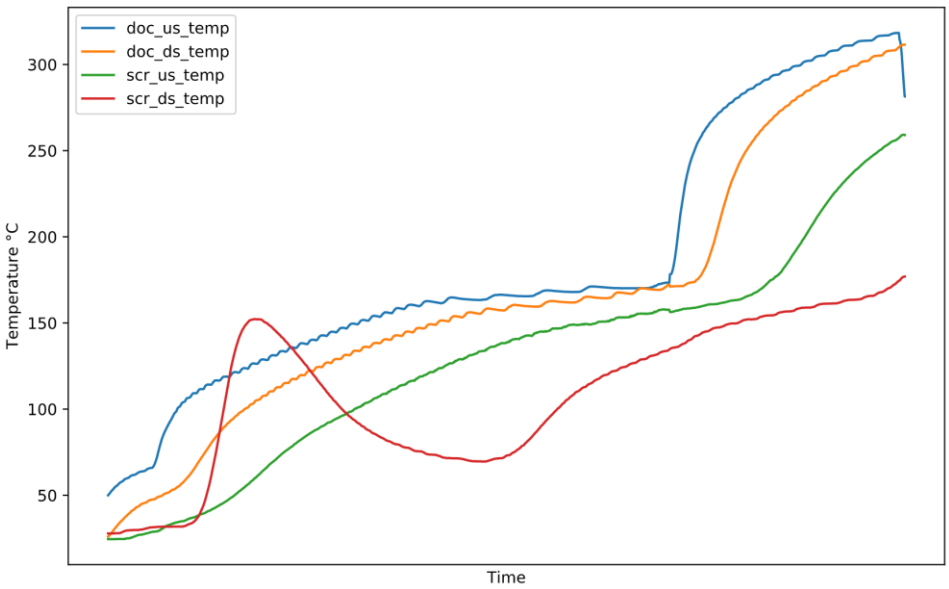

Figure 17 *Example temperature development in EAT*

As can be seen from Figure 17, the SCR downstream temperature doesn't follow a similar development as the other temperatures do. The other temperatures rise similarly as in the example shown in Chapter 3 but SCR downstream temperature falls after about 150 degrees and then it later starts to rise again. This problem occurs mainly with colder temperatures, but the test should work with all different preconditions.

More complex models can be created for predicting SCR temperatures [2] but different test starting conditions still make the model based solution more difficult. Machine learning is therefore applied for trying to make the model more accurate and capable for the specific use case.

4.3 Machine Learning Approach

Since the model-based approach doesn't fit very well for the specific problem, the selected approach is to train multiple machine learning models and compare their results. To further simplify the problem for a proof of concept, some limitations are set.

4.3.1 Limitations

The exhaust line consists of four different temperature sensor. In theory this will result to 24 possible sensor installation combinations in total, which is a factorial of 4.

$$n! = \prod_{i=1}^{n} i \tag{14}$$

Where *n* is 4.

In practice, not all these combinations are possible, due to the length of electrical cabling of the engine's wiring harness. Also some of the most obvious cases are displayed as faults with the ECU software. Therefore the number of possible combinations can be limited. In this thesis the number of combinations is also limited so that only sensors next to each other could be connected incorrectly. Therefore in this thesis there are four different possible combinations. Proof of concept can still be developed and the model would test the most difficult cases with just these selected combinations.

Agco Power manufactures multiple engine models. In the ideal solution the model would generalize so that the developed test would work with all the models. However since there are noticeable differences in the range of measurement values between the models and configurations, the scope of the research was limited to one engine model, an inline six cylinder (Figure 1).

4.4 Data

In order to create a functioning machine learning model, data is naturally required. This section explains the steps of the data collection process.

4.4.1 Data Collection

Inside the company there was already a lot of recorded field data from the engines. However, practically all the data was from engines where the sensors were installed correctly. This data could potentially be utilized, but the resulting model would only be able to classify the sensor connected either correctly or incorrectly. More detailed output could not easily be achieved.

For this study it was decided that new data would be generated specifically for this problem. Data collection was done using a laboratory engine. Data was collected during multiple measuring sessions and the result was 100 different recordings. The data was evenly distributed between the classes so that there are 25 samples of all of the different sensor combinations. Some of the recordings contain data from both the engine and DOC heatup phases and some contain data only from the DOC heatup.

Since the problem is a supervised classification problem, the data is labeled accordingly. Rather than labeling the data as either true or false, multi-labels are used. If for example the first two sensors are connected correctly and the other two incorrectly, measurement is labeled as [1,1,0,0].

4.4.2 Data Dimensionality Reduction

The recorded data consists of multiple recordings as described. One measurement session creates a time series of measurements signals. In the combined dataset there are then multiple measurement session instances. The length of each recording also differ from each other. In order to provide all the data as an input to the model, each of the two dimensional time series recordings need to be converted to one dimensional feature vectors. As a result the input data will be a two dimensional dataset with 100 feature vectors, matching the total number of recordings.

4.4.3 Feature Extraction

Current diesel engines have many sensors that are connected to them. This means that the number of used signals and variables must be limited so that only the essential data is given to the model. Irrelevant features will make the model perform weaker so only the ones best describing the data need to be preserved.

If some other sensors would be installed, more useful measurements could perhaps be achieved. However, if the system should only be tested with the diagnostics tool, no external sensors should be used. If more sensors would be attached, those would also have to be installed to every test engine and every time the test is performed. Naturally this is not wanted.

5. EXPERIMENTS

In this chapter the concrete process of creating the different models is described. The actual implementation is done using Python and scikit-learn [31] aka sklearn library.

5.1 Preprocessing

Since the data is specifically generated for this problem, not much preprocessing is needed. Some of the recordings contain missing values, and those are interpolated when creating the input to the model. The engine can have multiple states, e.g. starting and running. Only the states where the engine is heating up and the DOC is being heated are selected to the input data. After the features are created the data is then scaled using scikit-learn's standard deviation scaler before feeding it to the models.

5.2 Feature Creation

As described in the previous chapter, the features are derived from the recorded time series data. This requires calculating suitable values from the time series, e.g. mean and standard deviation. The sensor relations can also be utilized, for example dividing DOC upstream time series values with corresponding DOC downstream values and taking the mean or some other calculated value of that vector.

Another example of a generated feature is to take the measurement recording and fit a line to it and then taking the slope as a feature. The fit may not be entirely accurate but it can be an effective feature to the model. An example of a fitted line is shown in Figure 18.

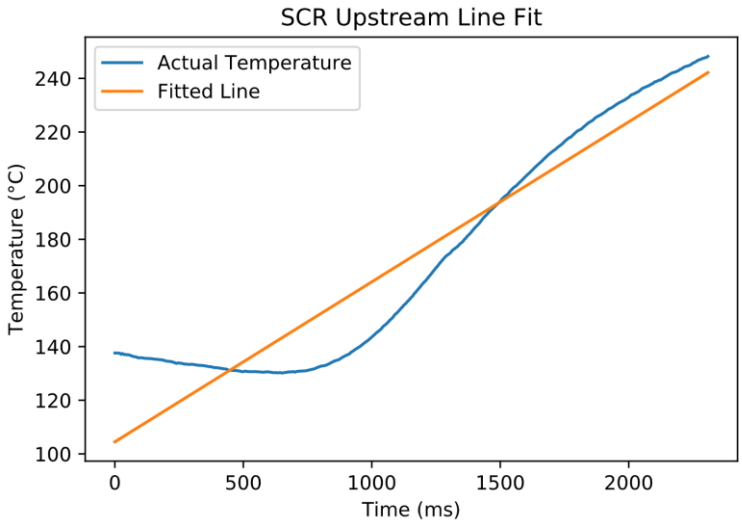

Figure 18 *First degree polynomial fit to measurement change*

In Chapter 3 the Savitzky-Golay filter was presented. The SG-filter is also used here to filter the temperature signals. In Figure 19 there's a representation of a SG-filter fit with a first order polynomial fit.

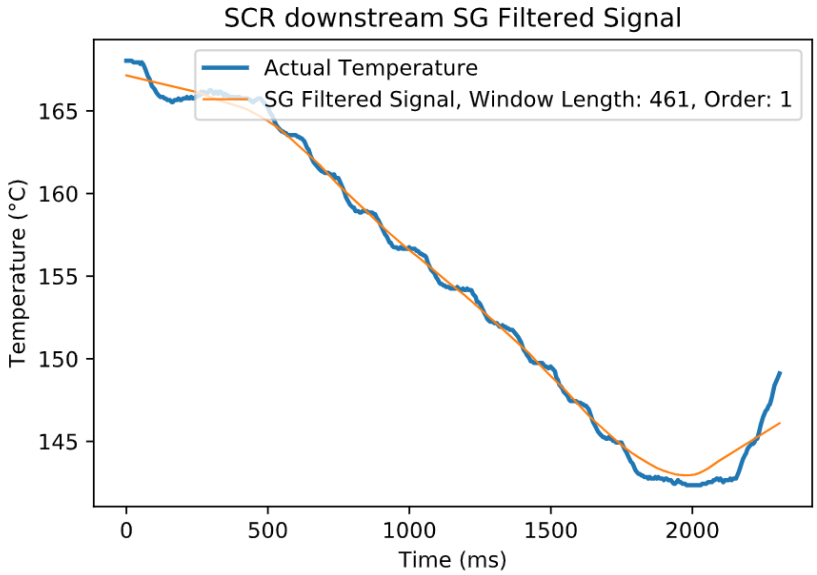

Figure 19 *First order Savgol filtered signal*

In this case a more accurate fit is achieved by using a second order polynomial. The fit can be altered with the order and the size of the convolution window. Example of a second order polynomial is displayed in Figure 20.

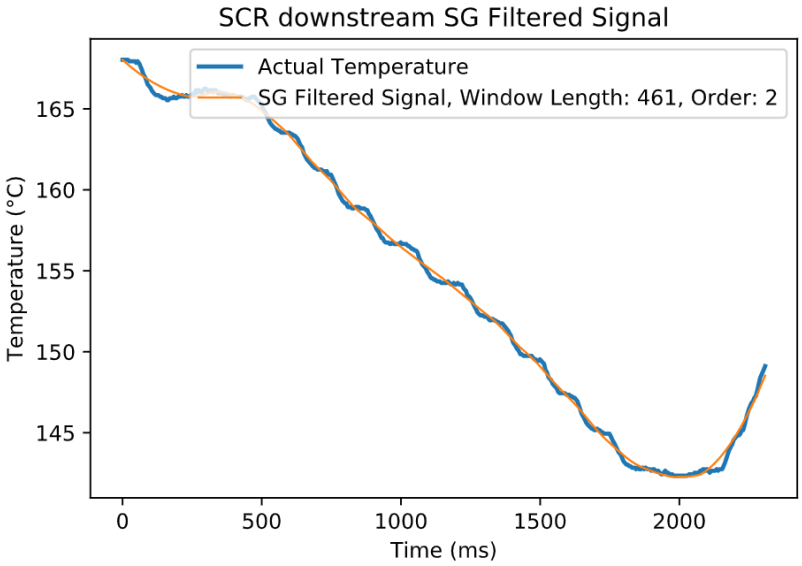

Figure 20 *Second order Savgol filtered signal*

As can be seen from the figures, the SG-filter accurately smoothens the curve. In this thesis the best fitting line is not necessarily what's needed but rather a line that best describes the data to the model. There are also other ways to extract information about the measurement data.

As described in Chapter 3, Savgol filter can also produce a derivative version of the signal. This can be a powerful tool for feature extraction. Figure 21 presents an example of an SCR downstream signal filtered with a derivative SG-filter.

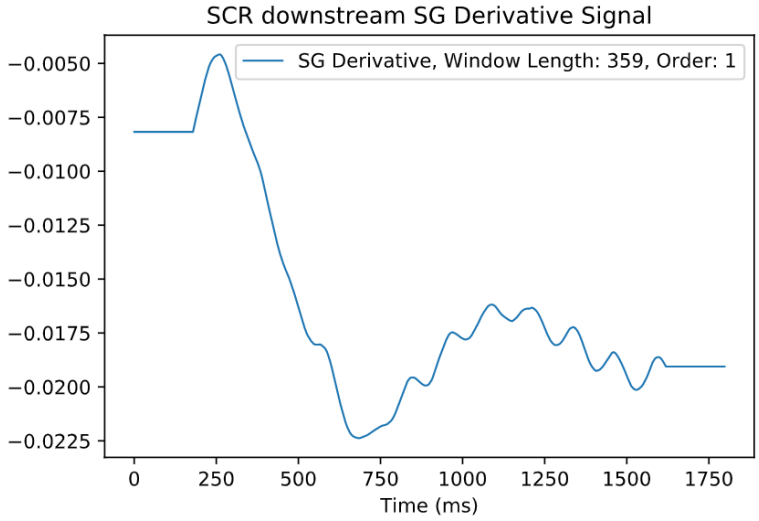

Figure 21 *Savgol derivative filter*

As can be seen from Figure 21, the values of the derivatives are not great in magnitude in this case. However, the filtered signals greatly differ from each other in shape. Figure 22 presents an example of similarly filtered signal from another sensor from the same recording.

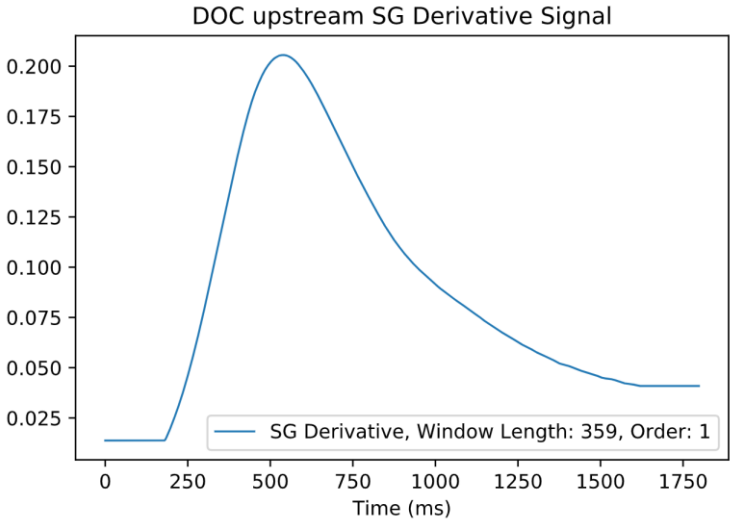

Figure 22 *Savgol derivative filter, example 2*

The two examples have almost the opposite behavior. The first one starts decreasing while the other one starts to decrease in magnitude. Also the values of the derivatives are higher. The two signals have noticeably different mean and standard deviation values. The differences therefore make them useful as features.

The features in this thesis are created using Savitzky-Golay filtered signals, Savitzky-Golay derivative filtered signals and line-fitted signals. During initial testing different combinations were tested. The final selected features are displayed in Table 1.

Table 1 *Used signals and functions applied to them*

Feature name	Feature description
scr_us_max	Max value of SCR upstream sensor
doc_us_savgol_mean	Mean of SG filtered DOC upstream
doc_ds_savgol_mean	Mean of SG filtered DOC upstream
doc_us/ds_savgol mean	Mean of SG filtered DOC upstream divided by DOC downstream
doc_ds/scr_us_savgol_mean	Mean of SG filtered DOC downstream divided by SCR upstream
doc_us/ds_savgol_max	Max value of DOC upstream divided by DOC downstream
doc_scr_us_savgol_max	Max value of DOC downstream divided by SCR upstream
doc_ds_derivative_mean	Mean of SG derivative filtered DOC downstream
scr_us_derivative_mean	Mean of SG derivative filtered SCR upstream
scr_ds_derivative_mean	Mean of SG derivative filtered SCR downstream
doc_ds_derivative_std	Standard deviation of SG derivative filtered DOC downstream
scr_us_derivative_std	Standard deviation of SG derivative filtered SCR upstream
scr_ds_derivative_std	Standard deviation of SG derivative filtered SCR downstream
doc_ds_curve_dir	Slope of a line fitted to the DOC downstream values
scr_ds_curve_dir	Slope of a line fitted to the SCR downstream values

All of the signals are Savgol filtered excluding the SCR upstream max value and the line fit slopes. During initial tests the filtered signals seemed to work better in general. From the line fit the slop is taken as a single value as described earlier. For the SG filtered signals max, mean, and standard deviation functions are applied to condense the signal to one value.

As can be seen from Table 1, only exhaust temperature sensor related signals are used. Another signals e.g. coolant temperature or oil temperature could be added. However, based on initial testing using those seems to have little to no effect. Therefore only the exhaust temperatures and signals derived from them are used and they should provide enough information for creating the models.

5.3 Used Models

There are multiple models in scikit-learn that can be used for this problem. There are implementations for the classifiers presented in Chapter 3 and those were chosen for training and evaluating in this thesis.

Table 2 *Used Models*

Short Name	Model
DEC_TREE	Decision Tree
RF	Random Forest
EXTTREE	Extremely Randomized Trees
KNN	K-Nearest-Neighbor
SVC_LIN	SVM, Linear
SVC_RBF	SVM, RBF
SVC_SIG	SVM, Sigmoid
LOG_REG	Logistic Regression

SVC stands for support vector classifier in sklearn, meaning it's a support vector machine classifier. Three different SVM kernels were selected.

5.4 Model Training

As described in Chapter 3, a preferable way to train a model is by using cross validation. In this thesis, cross validation is applied for every tested model during training and those results are then compared between each other to find the best overall model.

Randomly splitting the data to different cross validation folds may provide too optimistic results. In order to provide more strict splitting for training and testing, scikit-learn's GroupKFold [32] splitting is also used. GroupKFold works by giving a matching group for each sample in the data. In this case a group is one measurement session, which con-

tains four different recordings. As a result, GroupKFold will split the data so that recordings in the same group will not appear in different folds. Using this method, the model is not tested with data from the same measurement session what it was trained with. Recordings from the same measurement session may be more close to each other than those taken on another day. By splitting the data as described the predicting is more challenging for the classifier and overfitting can be avoided easier when training the model.

In sklearn the classifiers contain hyper parameters that can be changed. In order to get the best out of each model, different combinations of those hyper parameters are tested. This is done using the GridSearchCV [33] functionality in sklearn. GridSearchCV uses cross validation and tests different parameter combinations to find the one that gives the best score for the model. Those parameters can then be applied to the final model.

6. RESULTS

6.1 Feature Importances

Finding useful calculated features is somewhat an exploratory process. The models were trained multiple times with different features and the ones that didn't seem to have much effect were dropped. In scikit-learn, both random forest and linear regression classifiers can describe the importances of the features and they were also utilized. Figures 23 and 24 represent their results.

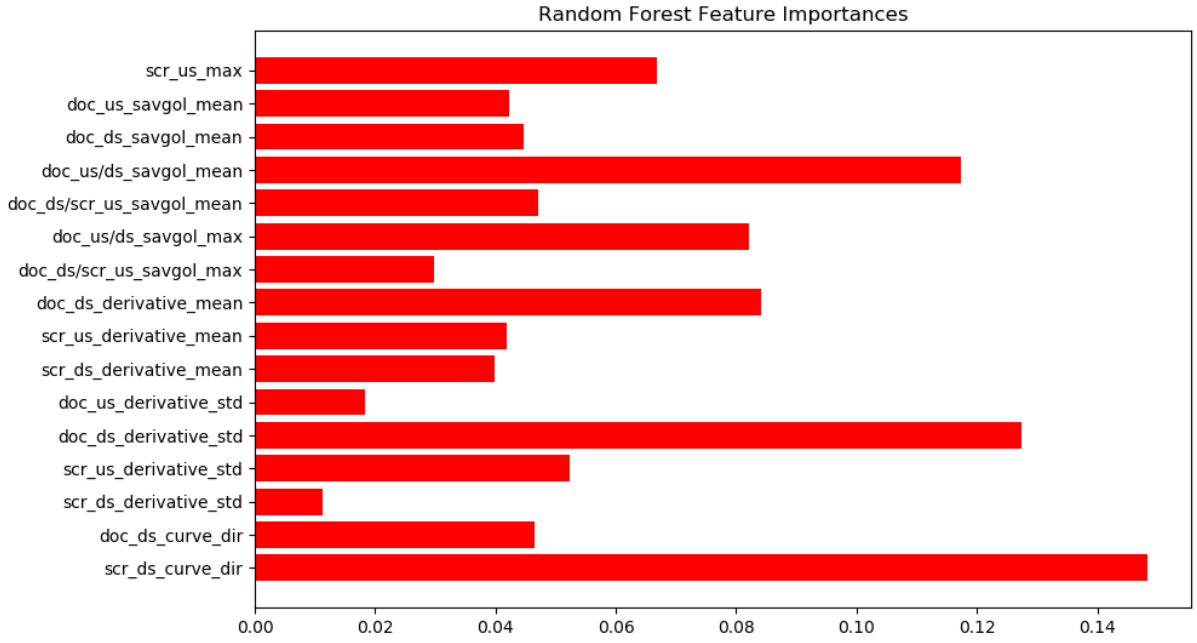

Figure 23 *Random Forest feature importances*

Random forest gives great emphasis for the SCR downstream line slope feature. Also the standard deviation for derivative filtered DOC downstream signal seems to be effective. Somewhat surprisingly the mean of regular Savgol filtered DOC signals divided by each other is also ranked highly. Other features are more close to each other in terms of ranked importance. The two features that are least favorable according to the algorithm are standard deviations of SG filtered DOC upstream and SCR downstream signals.

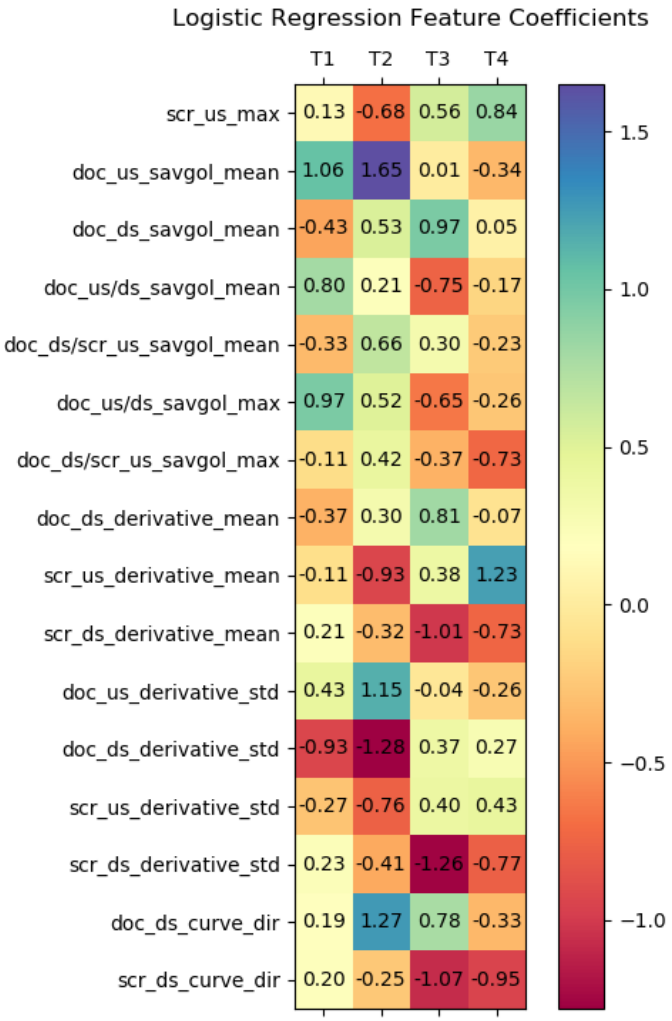

Figure 24 *Logistic regression coefficients*

Logistic regression algorithm ranks the features based on the weights that the classifier gives them. The importance is measured with the absolute values, the weight close to 0 carries little effect while the ones above 1 or below -1 in this case carry great value to the model. The horizontal axis specifies the weights for each sensor in the order of their correct placement in the system. Therefore T1 stands for DOC upstream sensor and T4 stands for SCR downstream sensor etc.

For logistic regression the most important feature is the mean of regularly SG filtered DOC upstream signal. Like for random forest, the standard deviation for derivative filtered DOC downstream signal is also weighted greatly. Interestingly however logistic regression also greatly favors the standard deviation of SG filtered SCR downstream sensor signal. This is the same feature that random forest ranked the lowest. Otherwise the rankings are somewhat close to each other.

6.2 Randomly Split Results

A simple way to test the models performance is to randomly divide the data to training and testing data. The model is then trained with the training data and performance metrics are evaluated with the testing data. The validity is increased by repeating the process multiple times and then averaging the results. Table 3 displays the results for 50 iterations test data consisting of 33% of all the data.

Table 3 *Randomly split data evaluation metrics*

Classifier	Mean Accuracy	Standard Deviation of Accuracy	Precision	Recall
DEC_TREE	0.967	0.035	0.991	0.991
RF	0.979	0.022	0.989	0.989
EXTTREE	0.995	0.020	0.998	0.998
KNN	0.999	0.004	1.000	1.000
SVC_LIN	0.990	0.014	0.999	0.999
SVC_RBF	0.990	0.020	1.000	1.000
SVC_SIG	0.990	0.013	1.000	1.000
LOG_REG	0.994	0.012	1.000	1.000

As can be seen from Table 3, the accuracy is almost 100%. More valid and better evaluations can be achieved with cross validation as described in Chapter 3.

6.3 Cross Validation Results

As described in Chapter 5, each model was trained using the GroupKFold splitting in sklearn. Cross validation was done using 5 number of folds. The scores for each folds were then averaged to give the final score for each models. Those results are presented in Table 4.

Table 4 *Cross validation scores, GroupKFold*

Classifier	Mean Accuracy	Standard Deviation of Accuracy	Precision	Recall
DEC_TREE	0.970	0.060	1.000	0.979
RF	0.960	0.058	0.992	0.986
EXTTREE	0.980	0.040	1.000	0.993
KNN	1.000	0.000	1.000	1.000
SVC_LIN	0.990	0.020	1.000	0.996
SVC_RBF	0.990	0.020	1.000	0.996
SVC_SIG	0.990	0.020	1.000	0.996
LOG_REG	0.990	0.020	1.000	0.996

Cross validation was also done by dividing the samples randomly to the folds. This was also done using 5 number of folds. The results are displayed in Table 5.

Table 5 *Cross validation scores, random folds*

Classifier	Mean Accuracy	Standard Deviation of Accuracy	Precision	Recall
DEC_TREE	0.960	0.080	1.000	0.971
RF	0.960	0.058	0.991	0.986
EXTTREE	0.970	0.060	1.000	0.989
KNN	0.990	0.020	1.000	0.993
SVC_LIN	0.980	0.024	0.995	0.996
SVC_RBF	0.980	0.024	0.995	0.996
SVC_SIG	0.980	0.024	0.995	0.996
LOG_REG	0.990	0.020	1.000	0.996

As can be seen from the cross validation results both of them provide excellent results. Little surprisingly the GroupKFold split cross validation gives slightly better results, even though the difference is not really noticeable. Also as seen from Table 3, randomly dividing the data to training and testing data and performing a single fit gives better results. This is what was expected.

6.4 ROC Curves

The ROC curves for the classifiers are created using GroupKFold using the same folds as for the evaluations in Table 4. Like the metrics presented in the previous chapter, the curves are created from the mean values of the splits. Since the true and false positive rates are so high, distinctive observations cannot be extracted from the results. ROC curves for all the classifiers are displayed in Figure 25.

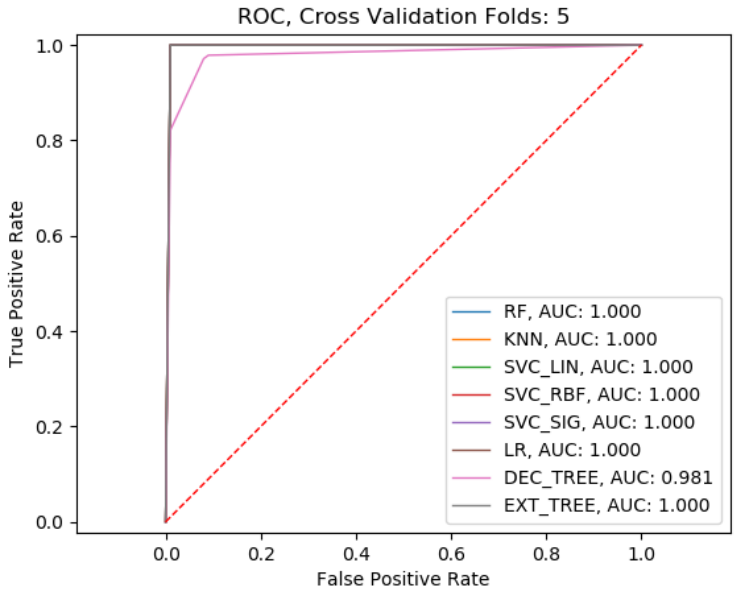

Figure 25 *ROC curves of the models*

Only the decision tree classifier ROC curve has lower AUC than the others. This further confirms that the classification abilities for all the models is nearly identical.

6.5 Real Life Test

Even though a model performs well in testing, the main goal is of course to have it function well in real life situations. In order to test this, an existing recording from a forestry machine was used as a real life example. The recording was taken with correctly installed sensors and warm engine.

The results are somewhat promising. All of the models except KNN and SVM with sigmoid kernel are able to correctly predict the validity of the sensor installations. The noticeable thing is that the used engine is different and the starting conditions differ from the ones used in training. However, in order to fully test the functionality, other combinations would need to be tested. Also testing should be expanded to other vehicle applications of the engine.

7. DISCUSSION

KNN classifier provides the overall best results in training. However, when tested with a real life recording it fails to correctly predict the sensor installations. Also as presented in the theory section, one of the main problems of the model is the overhead both in computing and model size. This is not a problem with small set of training data, but as the amount of data increases it may be a significant factor. This can matter specifically in this situation where the diagnostics tool bundle size is otherwise relatively small. Due to the challenges of KNN, logistic regression would probably be the best option since it seems to provide the best results overall in both training and real life testing.

The potential use cases of the model don't have to be limited with just testing the installation with the diagnostics tool. With modifications, the model can be extended to do plausibility checks for the temperature sensors when in use. There are two different use cases for this. The first case would be to extend the test functionalities to provide plausibility checks for individual sensors and their health. As it is currently, the model will look at the inputs given to it and then evaluate whether the input values seem to map to a correctly functioning sensor or not. Since the output is given separately to all of the sensors, they aren't dependent on each other even though that was the case in the original problem.

The other use case would be to evaluate the sensor health continuously in the engine software. In this case the model would be implemented to the engine ECU. If the sensor is faulty, a fault code could be given to the user of the vehicle displaying which of the sensor is perceived to be faulty. Even though the KNN classifier would gave good results it would most likely be useless in this case given the computational restrictions that were explained earlier.

With its current state, the model can be used for detecting possible incorrect sensor installations from a recording file. This can be useful with recordings that are received from customers. The recording can be given to the model and from that it can predict the sensor installations and then determine if some of them are connected incorrectly. This can be a useful engineering team tool for troubleshooting.

All of the alternative use cases require more data with different operating conditions and with different sensor installation combinations. However, in theory and based on the re-

sults achieved with this research the testing functionality could potentially be implemented. Future research could be done in this area, mainly focusing on the potential challenges for a production system.

8. CONCLUSION

The goal of the thesis was to research if a test system for verifying the correct EAT temperature sensor installation could be developed. This would require creating some sort of model for evaluating the plausibility of an individual temperature sensor. The research was done by first researching similar problems from literature and then creating a mathematical model. The problem was first approached with a simple first order system, but due to the challenges with it a machine learning solution was preferred. Data was collected by taking recordings from a test engine with different temperature sensor combinations. When the data was fully gathered, different machine learning models were tested to find the overall best performing one.

The first original research question was that is it possible to validate the correct installation of EAT system temperature sensors based on their measurements. Based on this research, the evidence suggests that the implementation of a described system is possible. The second research question was that if the implementation is possible in theory, how such a system could be developed in practice. The implementation in this thesis was tested with different machine learning models and all of those resulting models provide promising results. Therefore the training based solution might be the preferable approach.

The overall best model according to its accuracy and precision proved to be logistic regression. KNN classifier worked better in training, but it failed to classify with a real life test. Also due to possible computational and memory limitations other models might be preferred. Little surprisingly all of the models seemed to provide great results and there wasn’t major differences in terms of accuracy between them. This further suggests that more emphasis should be given to the data collection process.

However, in order to have a fully functioning model that works well in all situations, more development is required. As already mentioned, probably the most limiting factor with the current model is the amount of data that was used for training and testing it. Even though the achieved results are good, the models may not generalize enough in order to work flawlessly with other engines as well, which can be seen with the KNN classification results using a real life example.

The ambient conditions of the test remained almost the same during data collection and this may have an effect to the end result. Also the variance in the data could be added

by different starting temperature states in the recordings. Another thing to note is that in order to develop a model for a production version, data would also need to be generated from other sensor combinations that were not included now.

The created model is also trained with just one engine type. If all engines should work with the test, different models should be created for those. This would require collecting new data while the process would otherwise remain very similar.

To verify these assumptions more testing would be required. This could be a starting point for more extensive research. Another topic that could be researched more is the possibility to extend the model to do fault diagnosis on the engine ECU as presented in Chapter 7.

REFERENCES

[1] Abouel-Seoud S, Khalil M, Metwalley S, Allam E, Assad H. Consequence Analysis in Predictive Health Monitoring of Automotive Diesel Engine Defects. SAE International Journal of Engines 2013;6(3):1521-1531.

[2] Hu J, Wang J, Zeng J, Zhong X. Model-Based Temperature Sensor Fault Detection and Fault-Tolerant Control of Urea-Selective Catalyst Reduction Control Systems. Energies (Basel) 2018;11(7):1800.

[3] Agco Power. Agco Power HD 74. 2020; Available at: https://www.agcopower.com/products/engines/off-road-applications/hd-74/. Accessed Sep 9, 2020.

[4] The International Council Of Clean Transportation. European Stage V Non-Road Emission Standards. 2016; Available at: https://theicct.org/sites/default/files/publications/EU-Stage-V_policy%20update_ICCT_nov2016.pdf. Accessed Sep 9, 2020.

[5] DieselNet. Emission Standards: Europe: Nonroad Engines. 2016; . Accessed Oct 10, 2020.

[6] Agco Power. Agco Power Training Material. 2019.

[7] Ayodhya AS, Narayanappa KG. An overview of after-treatment systems for diesel engines. Environ Sci Pollut Res 2018;25(35):35034-35047.

[8] Pezzini A, Canova M, Onori S, Rizzoni G, Soliman A. A Methodology for Fault Diagnosis of Diesel NOx Aftertreatment Systems. IFAC Proceedings Volumes 2009;42(8)pp. 911-916.

[9] Canova M, Midlam-Mohler S, Pisu P, Soliman A. Model-based fault detection and isolation for a diesel lean NOx trap aftertreatment system. Control Eng Pract 2010;18(11):1307-1317.

[10] Gurung RB, Lindgren T, Boström H. Predicting NOx sensor failure in heavy duty trucks using histogram-based random forests. Int.J.Prognostics Health Manage. 2017;8(1):1-14.

[11] Chen R, Wang X. Model-Based Fault Diagnosis of Selective Catalytic Reduction Systems for Diesel Engines. SAE International Journal of Passenger Cars - Electronic and Electrical Systems 2014;7(2):449-453.

[12] Y. Wang, Y. Sun, C. Chang, Y. Hu. Model-Based Fault Detection and Fault-Tolerant Control of SCR Urea Injection Systems. IEEE Transactions on Vehicular Technology 2016;65(6)pp. 4645-4654.

[13] Hu J, Zeng J, Wei L. Failure diagnosis and tolerant control method for hydrothermally aged SCR system by utilizing EKF observer and MRAC controller. Energy 2018;156pp. 103-121.

[14] Guardiola C, Pla B, Piqueras P, Mora J, Lefebvre D. Model-based passive and active diagnostics strategies for diesel oxidation catalysts. Appl Therm Eng 2017;110:962-971.

[15] Skogestad S. Chemical and Energy Process Engineering. Baton Rouge: Taylor & Francis Group; 2008.

[16] Gollapudi S, Laxmikanth V. Practical Machine Learning. Birmingham, UK: Packt Publishing; 2016.

[17] Badillo S, Banfai B, Birzele F, Davydov II, Hutchinson L, Kam-Thong T, et al. An Introduction to Machine Learning. Clin Pharmacol Ther 2020;107(4):871-885.

[18] Gibaja E, Ventura S. A Tutorial on Multilabel Learning. ACM Computing Surveys (CSUR) 2015;47(3):1-38.

[19] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning data mining, inference, and prediction. 2nd ed. New York, NY: Springer New York; 2009.

[20] Scikit Learn. Nearest Neighbors Classification. Available at: https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py. Accessed Sep 22, 2020.

[21] Alpaydin E. Introduction to machine learning. Cambridge, Massachusetts: MIT Press; 2014.

[22] Zhang C, Ma Y. Ensemble Machine Learning Methods and Applications. 1st ed. New York, NY: Springer New York; 2012.

[23] Flach P. Machine Learning : The Art and Science of Algorithms that Make Sense of Data. Cambridge: Cambridge University Press; 2012.

[24] Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. Mach Learning 2006;63(1):3-42.

[25] Skansi S. Introduction to Deep Learning: from logical calculus to artificial intelligence. : Springer; 2018.

[26] scikit-learn. Receiver Operating Characteristic (ROC). 2020; Available at: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html. Accessed Oct 10, 2020.

[27] de Oliveira MA, Araujo NVS, da Silva RN, da Silva TI, Epaarachchi J. Use of Savitzky–Golay Filter for Performances Improvement of SHM Systems Based on Neural Networks and Distributed PZT Sensors. Sensors 2018;18(1):152.

[28] Wu JM, Tsai M, Huang YZ, Islam SH, Hassan MM, Alelaiwi A, et al. Applying an ensemble convolutional neural network with Savitzky–Golay filter to construct a phonocardiogram prediction model. Applied Soft Computing 2019;78:29-40.

[29] Savitzky A, Golay MJ. Smoothing and differentiation of data by simplified least squares procedures. Anal Chem 1964;36(8):1627-1639.

[30] Candan Ç, Inan H. A unified framework for derivation and implementation of Savitzky–Golay filters. Signal Process 2014;104:203-211.

[31] scikit-learn. scikit-learn: Machine Learning in Python. 2020; Available at: https://scikit-learn.org/. Accessed Oct 2, 2020.

[32] scikit-learn. GroupKFold. 2020; Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupKFold.html?highlight=groupkfold#sklearn.model_selection.GroupKFold. Accessed Oct 5, 2020.

[33] scikit-learn. GridSearchCV. 2020; Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Accessed Oct 5, 2020.